Research on Knowledge Assisted Software Engineering (KASE) and Development of a KASE Environment

Final Report

NASA Grant NCC 2-7494

Edward Feigenbaum, Co-Principal Investigator
Richard Fikes, Co-Principal Investigator
H. Penny Nii, Project Leader
Sanjay Bhansali, Research Associate

Knowledge Systems Laboratory
Department of Computer Science
Stanford University

Introduction

The following report summarizes the results of two years of research on Knowledge-Assisted Software Engineering (KASE) and the development of a KASE environment. The project was initiated on 1 November 1991, and was initially planned as a three-year effort. However, due to the loss of key personnel, and after consultation with the NASA project monitor, the project will terminate after two years.

Research Objectives

The goals of the research were: (1) to develop a software design environment that will enable software engineers to reuse extant software architectures and design knowledge, and thereby reduce the cost of software development as well as reduce errors in the software product; (2) to demonstrate the capability of the system and generality of the approach in different application areas including one NASA application.

Approach

Our approach consisted of the following steps.

(1) Develop a framework for synthesizing software using generic software architectures.

(2) Implement a set of domain-independent software tools to assist software designers in creating application specific software systems based on the framework.

(3) Acquire and represent generic architectures in different application areas as well as the relevant customization knowledge needed to customize the generic architecture for specific problem instances.

(4) Evaluate how synthesizing systems using the KASE approach compares with traditional methods.

Accomplishments

The specific research results to date have been well documented in several technical reports that appear in refereed conference proceedings and workshops (see bibliography); the work has thus been extensively peer-reviewed and widely disseminated. In this report the accomplishments are highlighted in bulletized form, with pointers to the literature.

Fiscal Year 1992

- Developed a general framework for designing software systems using generic architectures. The framework identified the major knowledge components and their representation, as well as the major processes involved in going from problem requirements to a system design.

- Used the framework to rationally reconstruct the design of two different systems, ELINT and HASP, using a common generic architecture and customization knowledge. This work was fully implemented and demonstrated the application-specific design support provided by KASE in a mixed-initiative and opportunistic design environment (Bhansali, 1992b; Bhansali & Nii, 1992a; Bhansali & Nii, 1992b). This work was presented and demonstrated to NASA personnel in May 1992.

- Extended the work on redesign support in KASE. Specific extensions include 1) a uniform representation mechanism for customization knowledge that provides plausible alternatives, default suggestions, and rationales for all design decisions (Bhansali, 1993); 2) dependency maintenance between design decisions allowing retraction of earlier design decisions with minimal impact on the rest of the design (Bhansali, 1992a); and 3) a scheme for providing flexible control for checking various kinds of constraints at any point in the design process (Nakano & Bhansali, 1993a).

- Enhanced the user interface, diagrammatic, and editing tools to facilitate the acquisition of domain models and the manipulation of an architectural design. Began work on using KASE to provide intelligent support to perform maintenance and future enhancements of KASE's own user interface.

- Identified a new domain, concerned with the analysis of radio occultation data received from planetary spacecraft, for application of KASE ideas. Finished representing the architectural design of a radio occultation data analysis system. Began work on 1) adding customization knowledge for instantiating parts of the architectural design to conform to different requirements; 2) adding transformation rules to convert certain classes of problems (involving the use of the NAIF library routines) to executable code.

Fiscal Year 1993

- Demonstrated the use of KASE in maintaining the Diagram-Manager subsystem of KASE. (Bhansali, 1993). A generic architecture of the Diagram Manager was represented in KASE, and design rules were encoded for constructing new kinds of layout diagrams based on properties of the objects and relations being depicted. Using these design rules, new layout diagrams were created much more efficiently than before, and by members of the KASE project other than the assigned programmer.

- Acquired the relevant problem-class model (objects, relations, operation definitions) for a subset of the radio occultation data analysis architecture. In order to limit the scope of the project to reasonable bounds, a problem-class was chosen that involved the

computation of the frequency shift of an observed signal. Various problem requirements and design parameters that affect the solution to problems belonging to this class were identified and represented in KASE. This work was done in collaboration with Joe Twicken, a member of the Radio Science group, EE department, Stanford University.

- A set of transformation rules were written in *Mathematica* to convert a high-level specification of problems belonging to the above problem class into an intermediate language. The transformation rules use information about problem requirements and design parameters during refinement.

- A translator that converts the intermediate language into FORTRAN was implemented.

- The system was used to generate a solution for a problem instance belonging to the chosen problem class. The solution obtained using KASE was compared with the solution written by a human programmer (Joe Twicken). The results of the comparison showed that:

  (1) the accuracy of solution generated by KASE is comparable to that produced by the human (the values computed by KASE were off by about 0.0018 %)

  (2) the length of the program produced by KASE was considerably more than that of the human,

  (3) the code produced by KASE was slightly less efficient than that of the human.

  The second and third results are related and can be easily improved by implementing a set of domain-independent optimizations of the code produced (e.g. replacing two or more subroutine calls by one). The results of this work have not yet been published, but will be described in a forthcoming technical report.

- Completed implementation of the Constraint-Checker subsystem of KASE begun the previous year. The constraint-checker has been empirically evaluated on existing and new software architectures. Details of the constraint-checking algorithm have been reported elsewhere (Bhansali, 1993; Nakano & Bhansali, 1993b).

Bibliography

Bhansali (1993). Synthesizing software using generic architectures (submitted to *Automating Software Engineering* journal).

Bhansali, S. (1992a). Generic software architecture based redesign. In *AAAI Spring Symposium on Computational Considerations in Supporting Incremental Modification and Reuse*, (pp. 53-58). Stanford, CA:

Bhansali, S. (1992b). The KASE approach to domain-specific software systems. In *AAAI Workshop on Automating Software Design*, (pp. 11-15). San Jose, CA:

Bhansali, S. (1993). Architecture-driven Reuse of Code in KASE. In *Fifth International Conference on Software Engineering and Knowledge Engineering*. San Francisco Bay: Knowledge Systems Institute (KSL 93-35).

Bhansali, S., & Nii, H. P. (1992a). KASE: An integrated environment for software design. In *2nd International Conference on Artificial Intelligence in Design*, (pp. 371-389). Pittsburgh, PA (KSL 91-73)

Bhansali, S., & Nii, H. P. (1992b). Software Design by Reusing Architectures. In *7th Knowledge-Based Software Engineering Conference*, (pp. 100-109). McLean, Virginia: IEEE Computer Society Press. (KSL 92-38)

Nakano, G., & Bhansali, S. (1993a). Flexible control mechanism in a consistency maintenance system. In *IEEE Pacific Rim Conference on Communications, Computers, and Signal Processing*. Victoria, British Columbia, Canada (KSL 93-27)

Nakano, G., & Bhansali, S. (1993b). A knowledge-based approach for consistency checking mechanism in software design. In *Proceedings of the 6th Florida AI Research Symposium*, (pp. 157-165). Fr. Lauderdale, FL (KSL 93-26)